VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2020 Andrey Girin

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

SYSTÉM PRO SLEDOVÁNÍ OSOB
PERSON TRACKING SYSTEM

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Andrey Girin**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Jaroslav Hájek**
SUPERVISOR

BRNO 2020

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Telekomunikační a informační systémy**

Ústav telekomunikací

Student: Andrey Girin ***ID:*** 185442
Ročník: 3 ***Akademický rok:*** 2019/20

NÁZEV TÉMATU:

Systém pro sledování osob

POKYNY PRO VYPRACOVÁNÍ:

Navrhněte a realizujte systém provádějící sledování osob. Systém bude za pomocí modulů ESP32 identifikovat osoby v místnostech a odesílat jejich polohu do interaktivního návrhu bytu či budovy ve webové aplikaci. Webový server poběží na jednodeskovém počítači Raspberry Pi. Výstupem bakalářské práce bude hotové řešení zobrazující polohu uživatele na základě měřených dat z modulů ESP32.

DOPORUČENÁ LITERATURA:

[1] TOLLERVEY, Nicholas H., [2018]. Programming with MicroPython: embedded programming with microcontrollers and Python. Sebastopol, CA: O'Reilly. ISBN 978-1491972731.

[2] LOMBARDI, Andrew, 2015. Websocket. Sebastopol, CA: O'Reilly. ISBN 978-1449369279.

Termín zadání: 3.2.2020 ***Termín odevzdání:*** 8.6.2020

Vedoucí práce: Ing. Jaroslav Hájek

prof. Ing. Jiří Mišurec, CSc.
předseda rady studijního programu

ABSTRAKT

Bakalářská práce se zabývá systémem provádějícím sledovaní osob za pomocí modulů ESP32. Vytvořený systém je schopen v reálném čase detekovat osoby za pomoci Bluetooth low energy na základě střední hodnoty RSSI. Systém se sestává ze tří hlavních bloků: aplikace pro modul ESP32, serverová aplikace, webové rozhraní. Vytvořená aplikace pro modulů ESP32 shromažďuje data BLE. Serverová aplikace zajišťuje komunikaci s postgreSQL databázi pro osobní identifikaci a poskytování API. Webové rozhraní slouží jako sledovací systém, který zobrazuje polohu osoby v místnosti a komunikuje s databází.

KLÍČOVÁ SLOVA

Chytrá domácnost, Flask, Python, IoT, Bluetooth low energy, MQTT, Websockets, ESP32, Raspberry pi.

ABSTRACT

The bachelor's thesis deals with a system that monitors people using ESP32 modules. The created system is able to detect people in real time using Bluetooth low energy based on the mean RSSI value. The system consists of three main blocks: application for ESP32 module, server application, web interface. The created application for ESP32 modules collects BLE data. The server application provides communication with the postgreSQL database for personal identification and API provisioning. The web interface serves as a tracking system that displays the person's location in the room and for communication with the database.

KEYWORDS

Smart Home, Flask, Python, IoT, Bluetooth low energy, MQTT, Websockets, ESP32, Raspberry pi.

GIRIN, Andrey. *Systém pro sledování osob*. Brno, Rok, 56 s. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce: Ing. Jaroslav HÁJEK

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Systém pro sledování osob“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu Ing. Jaroslav Hájek. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů

Úvod

Tato bakalářská práce se věnuje tématu chytré domácnosti, která si rychle získává na popularitě. Dále se zabývá fenoménem interakce uživatelů uvnitř prostoru, a to poskytováním služeb uživateli na základě jeho polohy (LBS - Location-based service). Díky údajům o uživateli a jeho poloze je možné vytvořit individuální scénář chytré domácnosti, který bude založen na osobních preferencích uživatele. Kromě toho lze řešení použitá v tomto projektu aplikovat ve vnitřní navigaci (indoor navigation) pro orientaci a interakci ve vnitřním ekosystému, a to nejen rodinných domů, ale také velkých objektů jako jsou letiště, muzea nebo nákupní centra. Klíčovým aspektem projektu je systém, pomocí kterého je možné přesně sledovat polohu uživatele v prostoru a identifikovat jeho osobu.

Systém je napsán v programovacím jazyce Python. Ke sběru dat využívají chytré domácnosti protokol MQTT(Message Queuing Telemetry Transport) a architekturu rozhraní REST(Representational State Transfer). Program shromažďuje, zpracovává a publikuje data v reálném čase. Na základě zpracovaných dat jsou interaktivně zobrazena umístění uživatele a jejich změny v prostoru. Grafické prvky programu jsou vytvořeny pomocí HTML(Hypertext Markup Language) a CSS(Cascading Style Sheets), které umožňují zobrazit program v prohlížeči. Každý uživatel může do seznamu přidat své vlastní zařízení pomocí rozhraní API(Application Programming Interface). K přidání zařízení stačí znát jeho adresu MAC(Media Access Control). Jako server pro tento program lze použít jednodeskový počítač Raspberry Pi. V projektu je detekce prováděna pomocí BLE Bluetooth majáků, které představuje mikrokontrolér ESP32. Majáky jsou instalovány po obvodu místnosti a se specifikovanou frekvencí vysílají signál obsahující informaci, která uživatele identifikuje. Aplikace určuje souřadnice uživatele na základě síly signálu jeho zařízení, přičemž tímto zařízením může být i fitness náramek.

.

1 Chytrá domácnost

V této kapitole je popsána technologie Bluetooth Low Energy a zejména kanály, struktura paketů, technická charakteristika a rozdíl mezi ním a klasickým Bluetooth. Kapitola se také zabývá použitím této technologie v projektu pro detekci osoby.

1.1 IoT (Internet of Thigs)

IoT je síť vzájemně propojených výpočetních zařízení, pomocí kterých je možné shromažďovat informace, analyzovat je a vytvářet akce. Těmito zařízeními mohou být **senzory**, **teploměry**, mikrokontroléry a další. Každé zařízení je vybaveno UID (Unique identifier) a schopností přenášet data přes síť. K implementaci komunikace internetu věcí se používají různé komunikační technologie, s nimiž je možné přenášet data na velké vzdálenosti s nízkou spotřebou energie. Nejvýznamnějšími jsou **LoraWan** a **Sigfox** s velkým pokrytím až do 15 km. Tyto technologie pracují v nelicencovaném frekvenčním pásmu 868 MHz, ale nemají velkou přenosovou rychlost. Tyto technologie se používají v případě, že je nutné získat data ze senzorů v krátkém časovém úseku, po zbytek času jsou v režimu spánku. Další možností je technologie přenosu dat Bluetooth, která umožňuje zařízením komunikovat na vzdálenost až 100 metrů a zároveň má dobrou rychlost přenosu dat. Pro internet věcí se používá možnost Bluetooth Low Energy.[2]

1.2 Bluetooth

Bluetooth je bezdrátová PAN (Personal Area Network) technologie, která propojuje dvě a více elektronických zařízení na krátké vzdálenosti pomocí rádiových vln (Ultra High Frequency) v pásmu 2,400 až 2,483 5 GHz.

Bluetooth byl vytvořen v roce 1994 firmou Ericsson a byl koncipován jako bezdrátová alternativa k datovým kabelům RS–232. Od roku 2002 patří technologie Bluetooth ke standardu norem IEEE 802.15.1 (bezdrátová PAN sít).[3]

1.3 Bluetooth Low Energy

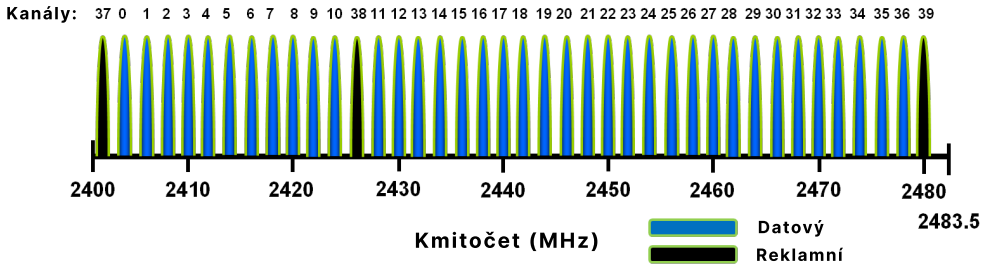
BLE (Bluetooth Low Energy) byl vyvinut na základě Bluetooth verze 4.0. Jeho výhodou je snížení spotřeby energie ve srovnání s Bluetooth, přitom má podobný bezdrátový dosah

Spotřeba energie je u BLE udržována na minimální hodnotě tím, že připojené zařízení se po většinu času nachází ve stavu spánku a probouzí se jen tehdy, kdy

je inicializována jeho aktivní komunikace. Zároveň tato komunikace trvá jen několik milisekund. Průměrný odběr proudu závisí na frekvenci požadavků na komunikaci a typicky činí méně než 1 µA. Z toho vyplývá, že pro napájení zařízení po dobu jednoho roku a více postačí knoflíková baterie.

1.3.1 Frekvencni pasmo BLE

Technologie Bluetooth Low Energy pracuje v rozsahu 2,4 až 2,483$\dot{5}$ GHz jako klasické Bluetooth, ale používá jinou sadu kanálů, která umožňuje zařízením s duálním režimem sdílet jednu rádiovou anténu. Místo klasických kanálů Bluetooth (79 kanálů po 1 MHz) technologie Bluetooth Low Energy rozděluje toto pásmo na 40 kanálů při 2 MHz rozestupu od 2 400 do 2 483,5 MHz, počínaje od 2$\dot{4}$02 MHz (příklad na obr. 1.1).



Obr. 1.1: Bluetooth Low Energy kanály

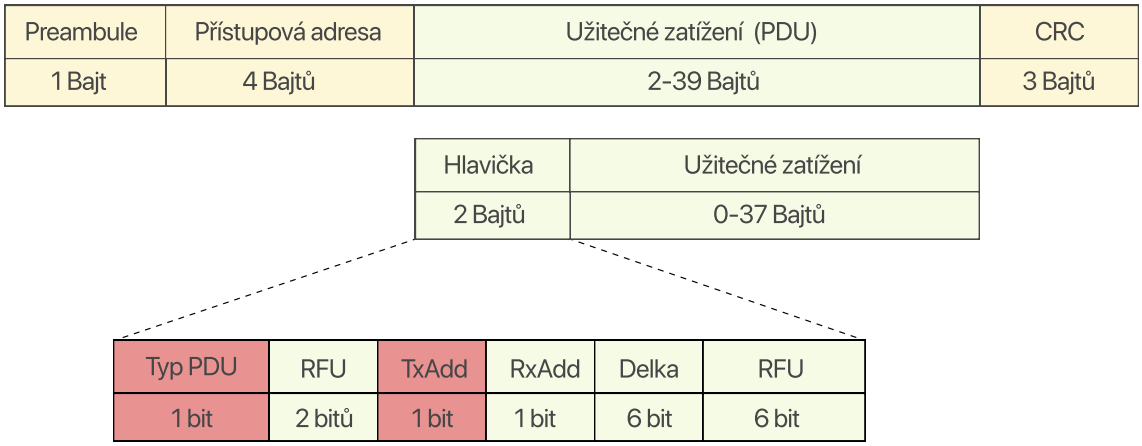
Bluetooth Low Energy existují 2 typy kanálů: reklamní kanál (*Advertising channels*) a datový kanál (*Data channels*). Reklamní kanál používá 3 kanály, které jsou určeny pro zjišťování zařízení, navázání spojení a vysílací přenosy. Jedná se o 37 (2 402 MHz), 38 (2 426 MHz) a 39 (2 480 MHz) kanálů. Díky tomuto uspořádání reklamních kanálů je zařízení i při nízké úrovni výkonu schopno přijímat signál, protože jsou umístěna mezi kanály Wi-Fi (1, 6, 11). Datový kanál používá 37 kanálů a slouží pro obousměrnou komunikaci mezi připojenými zařízeními a používá adaptivní přeskakování frekvence (Adaptive Frequency Hopping) pro následné připojení. Adaptivní přeskakování frekvence je prováděno pouze na 37 datových kanálech a řídí se vzorcem **??**

$$f_{n+1} = (f_n + hop) \bmod 37 \tag{1.1}$$

Kde $f_n + 1$ je frekvence (kanál), která se použije při příštím připojení, a *hop* je hodnota, která se může pohybovat v rozmezí 5–16 a je nastavena při vytvoření spojení. Modul 37 odpovídá počtu datových kanálů.[4]

1.3.2 Struktura paketu BLE

Struktura paketu BLE je stejná pro reklamní i datové kanály, s výjimkou PDU (Protocol Data Unit), jehož obsah a objem se liší. Pro detekci využívá reklamní kanál a má dva cíle: vysílat data pro aplikace, které nevyžadují úplné připojení, a najít pomocné zařízení a připojit se k němu. Paket Bluetooth Low Energy se skládá z několika částí, které jsou znázorněny na obrázku[1] 1.2.



Obr. 1.2: Struktura paketu BLE [1]

Preambule - má jednobajtovou hodnotu, která se používá k synchronizaci s přijímači. U vysílacích paketů je hodnota „0xAA“.

Přístupová adresa má pevnou hodnotu 4 bajty a slouží k zajištění toho, aby zařízení pochopila, pro koho je paket BLE určen. Pokud zařízení nezná tento kód, paket bude ignorován. Díky tomuto kódu, který je „0x8E89BED6““, se všechna zařízení na reklamních kanálech navzájem vidí.

Užitečné zatížení paketu(PDU) sestává ze záhlaví užitečného zatížení.

- **Záhlaví** popisuje typ paketu a typ PDU určuje účel zařízení. Nejdůležitějšími prvky záhlaví jsou:
  - Existují následující typy reklamních paketů (viz tabulka 1.1).
    ADV_IND – jedná se o nepřímé pakety, které odesílají zařízení připravená k připojení. Zařízení se používá k rozesílání reklamních paketů.
    ADV_DIRECT_IND – cílené reklamní pakety pro připojená zařízení. Připojit se k nim může pouze konkrétní zařízení s dříve známou MAC adresou.
    ADV_NONCONN_IND – reklamní pakety, které odesílají informace o zařízení, ale nepřipojí se.
    SCAN_REQ, SCAN_RSP, CONNECT_REQ – pakety vyměněné připojenými zařízeními během navazování synchronního připojení. .

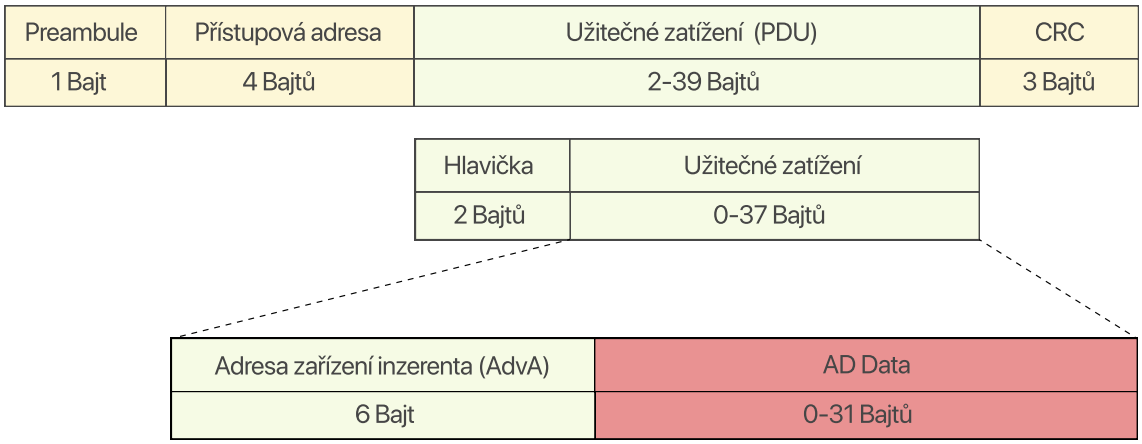
Typ PDU	Název paketu
0000	ADV_IND
0001	ADV_DIRECT_IND
0010	ADV_NONCONN_IND
0011	SCAN_REQ
0100	SCAN_RSP
0101	CONNECT_REQ
0110	ADV_SCAN_IND
0111-1111	Reserved

Tab. 1.1: Typy reklamních paketů

ADV_SCAN_IND – pakety odesílány nepřipojitelným zařízením, které může poskytnout další informace v reakci na požadavek na skenování.

- **TxAdd** označuje, jaká bude MAC adresa zařízení. Pokud `TxAdd = 0` MAC adresa je veřejná, pokud `TxAdd = 1` MAC adresa je náhodná.

- **Užitečné zatížení** paketu zahrnuje MAC adresu zařízení (AdvA – Advertiser device address) spolu s uživateli definovanými daty zahrnutými do uživatelem definovaného inzerovaného datového paketu, jak je znázorněno na obrázku 1.3. Tato pole jsou adresa a data vysílaného majáku.

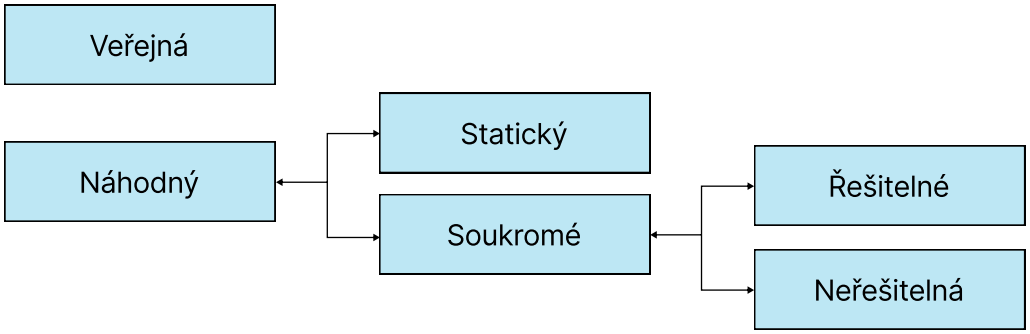


Obr. 1.3: Užitečné zatížení struktura

Cyklický redundantní kód (CRC) je poslední částí přenášeného paketu. Cyklická kontrola nadbytečnosti je kód detekující chyby, který se používá ke kontrole integrity paketu proti nežádoucím změnám, obvykle kvůli ruchu ve vzduchu. Tím je zajištěna integrita dat všech paketů přenášených vzduchem.[1]

1.3.3 Struktura užitečného zatížení

Adresa zařízení (AdvA) může být veřejná nebo náhodná. Veřejná adresa se používá jako jedinečný identifikátor pro organizaci OUI (Organizationally Unique Identifier). Náhodnou adresu generuje zařízení. Generovaná adresa může být tří různých typů: statická, řešitelná soukromá (*resolvable private*) a neřešitelná soukromá (*nonresolvable private*) (obr 1.4).[5]



Obr. 1.4: Typy adres zařízení

Dokud se zařízení restartuje, není dovoleno měnit statickou adresu. Soukromá adresa se může časem změnit. Neřešitelná adresa se taktéž může časem změnit, což je její rozdíl oproti adrese statické. K získání skutečné adresy lze použít řešitelnou adresu. Náhodná adresa je nástroj pro ochranu osobních údajů, který zabraňuje sledování zařízení.

Typ dat AD	Hodnota	Popis
Flags	0x01	Možnosti zjišťování zařízení
UUID Service	0x02...0x07	Služby GATT-zařízení
Lokální název	0x08...0x09	Název zařízení
TX Power Level	0x0A	Výstupní výkon zařízení
Údaje specifické pro výrobce (Manufacturer-Specific Data)	0xFF	Definováno uživatelem

Tab. 1.2: Typy dat reklam (AD)

[1]

1.4 Parametry Bluetooth oproti BLE

Bluetooth Low Energy a Bluetooth verze jsou zaměřeny na bezdrátový přenos dat, ale parametry BLE se liší oproti klasické technologii Bluetooth. Přehled hlavních rozdílů je uveden v tabulce 1.3.

Tab. 1.3: Porovnání technologie Bluetooth a Bluetooth Low Energy.

Technické specifikace	Bluetooth	Bluetooth Low Energy
Vzdálenost	100 m	50 m
Maximální propustnost	1–3 Mbit/s	1 Mbit/s
Typická propustnost	0,7–2,1 Mbit /s	0.27 Mbit/s
Max. počet slave zařízení	7	Není definovaný. Závisí na implementaci
Robustnost	Adaptivní rychlé frekvenční přeskakování, FEC , rychlé, ACK.	Adaptivní kmitočtové přeskakování, líné potvrzení, 24bitový CRC, 32bitová kontrola integrity zpráv
Latence	Obvykle 100 ms	6 ms
Celkový čas na odeslání dat	100 ms	3 ms, méně než 3 ms
Spotřeba energie	1 W jako reference	0,01–0,50 W
Špičková spotřeba proud	Méně než 30 mA	Méně než 20 mA

2 Návrh hardware

V této kapitole jsou podrobněji popsány komponenty hardwaru, které jsou použity k realizaci projektu (Raspberry Pi, ESP32 a jejich technické charakteristiky, architektura a komunikace Raspberry Pi a ESP32 pro výměnu dat).

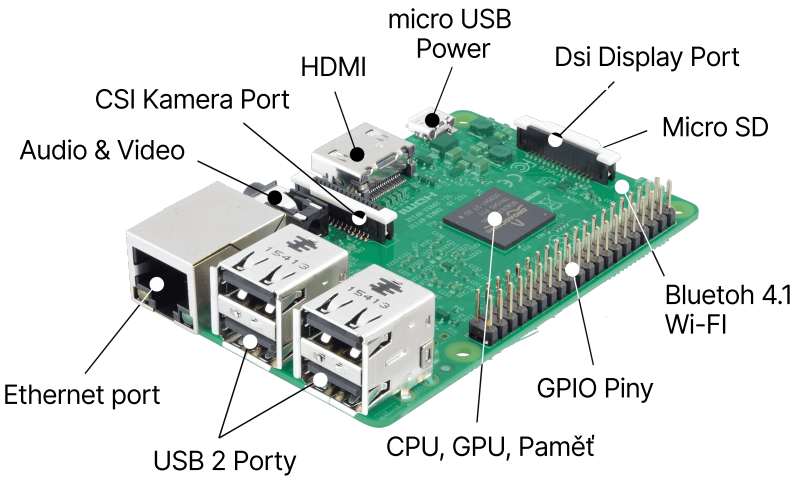
2.1 Raspberry Pi

Raspberry Pi je malý jednodeskový počítač s deskou plošných spojů o velikosti platební karty, který je možné použít jako běžný desktop počítač, pokud se k Raspberry Pi připojí monitor. Aktuálně Raspberry Pi nabízí řadu modelů od nejlevnějšího Rasberry Pi Zero po nejnovější Raspberry Pi 4 B. Jako základní hardware, na kterém bude umístěn webový server a který bude provádět komunikaci s modulem ESP32, je zvolen Raspberry Pi 3 B+, neboť splňuje technické parametry, které jsou nutné pro realizaci projektu, a zároveň je levnější než čtvrtá generace.[6]

2.1.1 Raspberry Pi model 3B +

Model Raspberry Pi 3 B+ vznikl v březnu roku 2018 a je finální verzí Pi řady 3. Počítač využívá architekturu ARMv8 (64/32bit), která je umístěna na 64bitovém čtyřjádrovém procesoru „Cortex-A53“ 1,4 GHz. Systém pracuje na čipu „Broadcom BCM2837B0“. Základní architektura „BCM2837B0“ je identická s čipem „BCM2837A0“ používaným v jiných verzích Pi. Základní hardware ARM (Advanced RISC Machine) je stejný, pouze jeho frekvence je vyšší. Čip „BCM2837B0“ je balen odlišně od „BCM2837A0“ a primárně lépe odvádí teplo. Pracuje na vyšších kmitočtech (z důvodu provozu při nižším napětí pro snížení spotřeby energie) a má přesnější kontrolu teploty čipu. Pi 3 B+ používá integrovanou 32bitovou grafickou kartu „Broadcom VideoCore-IV“, která funguje při 400 MHz a podporuje OpenGL ES 2,0, MPEG-2 a VC-1 (s licencí) 1080p30 H.264 / dekodér a enkodér vysokoprofilového MPEG-4 AVC. Paměť (SDRAM) s objemem 1 gigabyte je sdílena s grafickou kartou.

Raspberry Pi 3 B+ má port HDMI, gigabitový ethernet, 3,5 mm audio jack, 15-pin rozhraní „MIPI“, kamerový (CSI) konektor, který se používá s kamerou Raspberry Pi, MicroSD slot, 10/100 Mbit Ethernet, 17×GPIO (univerzální vstupní a výstupní pin), 5 V MicroUSB. Obrázek 2.1 zobrazuje porty. Raspberry Pi 3 B+ podporuje WiFi 802.11ac, Bluetooth verze 4.2 a BLE.



Obr. 2.1: Raspberry Pi 3 B+ porty

2.2 ESP32

ESP32 je mikrokontrolér vyrobený společnosti Espressif Systems. Je to vysoce integrovaný kombinovaný čip s Wi-Fi a Bluetooth technologií, využívající mikroprocesor „Tensilica Xtensa LX6“ ve dvoujádrových či jednojádrových variantách. Mikrokontrolér ESP32 je vyrobený pro řešení vyžadující minimální spotřebu energie.

Společnost Espressif nabízí celou řadu modulů ESP32. V projektu je použit model ESP32-WROOM-32D (viz obr. 2.2), protože podporuje technologii Bluetooth, na rozdíl od starších verzí ESP.[7]

Obr. 2.2: Užitečné zatížení struktura

2.2.1 ESP32-WROOM-32D

ESP32-WROOM-32D je univerzální modul s integrovaným jádrem mikrokontroléru a podporou komunikačních standardů Wi-Fi + BT + BLE, určený pro energeticky úsporné aplikace. Modul může přenést hlas, streamovat zvuk a MP3 audio. Modul ESP-WROOM-32D je založen na čipové sadě ESP32-D0WD. Čipová sada obsahuje dvou-jádrový procesor Xtensa®32-bit LX6 MCU, senzor ADC/DAC. Napájení 3.0 ~3.6 V, proud 80 mA, minimální hodnota proudu je 500 mA.

Klíčové parametry zařízení jsou:

- 240 MHz dvou-jádrový mikrokontrolér Tensilica LX6,
- Integrovaná 520 KB SRAM (Static Random Access Memory)
- 16 MB flash paměť CPU (Central Processing Unit)
- Integrovaný Wifi vysílač a přijímač podporující standardy IEEE 802.11 b/g/-n/e/i v pásmu 2,4 GHz,
- Možnost zabezpečení Wifi provozu pomocí WEP,WPA/WPA2 PSK/Enterprise,
- Integrovaný Bluetooth v4.2 BR/ EDR (Enhanced Data Rate)/ BLE,
- Zabudovaná PCB (Printed Circuit Board) anténa i IPEX konektor umožňující připojení externí antény,
- Napětí 3.0 V až 3.6 V,
- Provozní teplota −40°C až +105°,
- 32(34–36) GPIO (General-purpose input/output) pinů (Podporující různá rozhraní jako UART, SPI, I2C , LED, PWM atd. (některé piny obsahují A/D a D/A převodníky),
- Low-power Management (správa nízké spotřeby) umožňující přepínání mezi pěti různými módy spotřeby a tím výrazně zvýšuje dobu provozu. [8]

Napájeni by melo být pouze bateriové. Správně vybraná baterie dokáže poskytnout největší možnou kapacitu a zároveň potřebné napětí pro napájení zařízení. Modul ESP32 pracuje s napětím v rozmezí 3.0 až 3.6 V. Proto baterie musí splnit následující kritéria:

- velikost napětí,
- maximální hodnota dodávaného proudu,
- velikost klidového proudu,

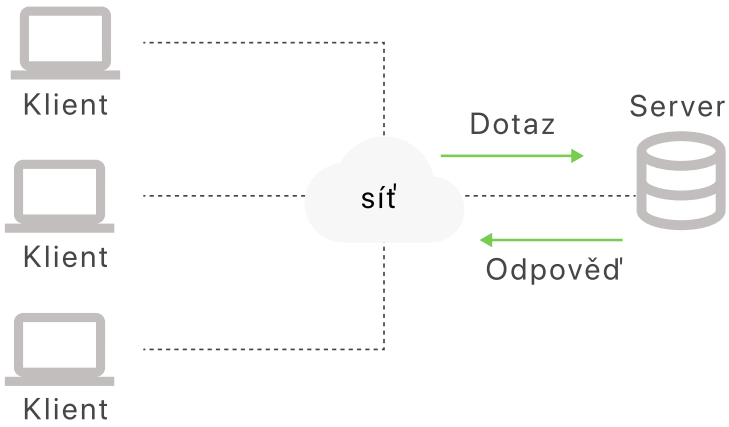
3 Architektura

Architektura aplikace je soubor rozhodnutí o organizaci systému ztělesněného v jeho prvcích, jejich vzájemných vztazích a prostředí, který zahrnuje:

- Speciálně vybranou sadu strukturálních prvků uspořádaných určitým způsobem pro vzájemnou interakci, pomocí které je systém složen.
- Soubor vybraných prvků jako součástí větších systémů.
- Architektonický styl pro uspořádání jednotného přístupu.

3.1 Architektura klient–server

Architektura klient–server je struktura, která distribuuje úkoly mezi klientem a serverem. Klient je obvykle aktivním prvkem, zatímco server je pasivní. Klient odešle požadavek a server odešle odpověď obr 3.1



Obr. 3.1: Architektura klient-server

3.1.1 Webová aplikace

Webová aplikace je aplikace typu klient-server, ve které je klientem prohlížeč a serverem je webový server. Klient přijímá data z webového serveru a aby uživatel viděl grafické rozhraní aplikace v okně prohlížeče, musí klient zpracovat přijatá data pomocí HTML(Hypertext Markup Language), CSS(Cascading Style Sheets) a JS(JavaScript) pro zobrazení v prohlížeči.

Webový server je server, který přijímá HTTP (Hypertext Transfer Protocol) požadavky od klientů a vydává na ně HTTP odpovědi. Webový server se označuje jako software, který funguje jako webový server a počítač, na kterém běží. Nejběžnější typy softwaru webového serveru jsou Apache, IIS a NGINX. Na webovém serveru

běží testovací aplikace, kterou lze implementovat v programovacím jazyce Python, Ruby, PHP, Java, Perl a dalších.

Databáze je informační model, který umožňuje uspořádat ukládání dat o objektu nebo skupině objektů, které mají sadu vlastností, které lze kategorizovat. Databáze fungují pod kontrolou tzv. systémů správy databází (dále jen DBMS). Nejoblíbenějšími DBMS jsou MySQL, MS SQL Server, PostgreSQL, Oracle (všechny jsou typu klient–server).

3.2 MVC

MVC (Model View Controller) je softwarová architektura, která rozděluje datový model aplikace, uživatelské rozhraní a řídicí logiku do tří nezávislých komponent tak, že modifikace některé z nich má jen minimální vliv na ostatní. Rozděluje aplikaci do těchto tří vzájemně propojených částí obr.3.2:

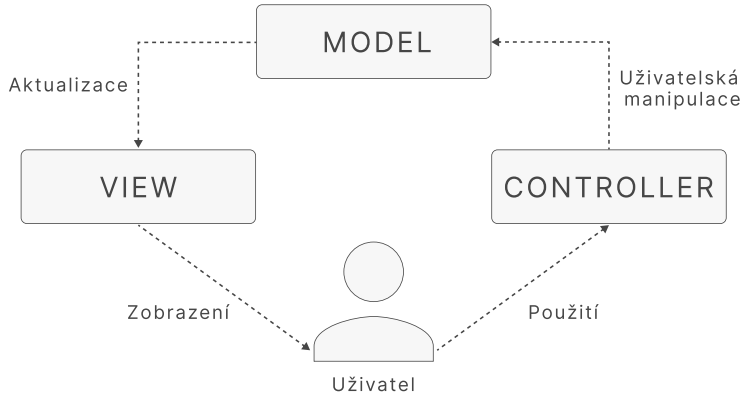
Model – Jedná se o aplikační data a logiku jejich příjmu a uložení. Toto je doménový model založený na databázi nebo na základě výsledků z webových služeb

- Zpracovává aplikační data
- Tvoří centrální část MVC.

View – Je zodpovědný za zobrazení uživatelského rozhraní na obrazovce. Bez knihoven widgetů to znamenalo vlastní vykreslování bloků, tlačítek, vstupních polí atd. Zobrazení také může monitorovat model a zobrazovat z něj data.

- Může to být jakákoli výstupní reprezentace informací pro uživatele.
- Vykresluje data z modelu do uživatelského rozhraní.

Controller Zpracovává akce prováděné uživatelem a aktualizuje **Model** nebo **View**. **Model** ukládá data, která se získávají podle příkazů z kontroléru. **View** generuje



Obr. 3.2: Model View Controller

výstupní informaci pro uživatele na základě změn v **Modelu**. **Controller** působí na

Model a **View**; odešle příkazy do **Modelu**, aby aktualizoval svůj stav, a do **View**, aby změnil informace prezentované uživatelům.[9]

3.3 REST (Representational State Transfer)

REST je architektonický styl pro aplikaci klient-server, která zjednodušuje směrování a vytváření API(Application programming interface) rozhraní pro programování aplikací. V RESTu dochází ke správě informací bez dalších vrstev. Každý objekt je identifikován globálním identifikátorem `URL`, kde každá adresa `URL` má přesně definovaný formát. Správa objektů přichází prostřednictvím stavových kódů HTTP(S), kde se v HTTP požadavky „GET“ nebo „POST“ nazývají „požadavek REST“.[10]

Informace jsou vyměňovány v RESTful API pomocí dokumentů JSON(JavaScript Object Notation) a XML(Extensible Markup Language), kde jsou nezbytná data předávána jako parametry požadavku. Typy požadavků REST jsou následující:

- **GET** je metoda HTTP požadavků na příjem definitivních informací ze strany serveru. Při použití tohoto požadavku server vyhledá potřebné informace (načtené na serveru) a odešle je zpět klientovi.[10]

Výpis 3.1: požadavek GET

```
curl -i -X GET https://PTS/api//user/xpgirin00
{
"id": 185442,
"jmeno": "Andrey",
"prijmeni":"Girin",
"username":"xpgirin00",
}
```

- **POST** je požadavek o vytvoření. Používá se k vytvoření konkrétního objektu na serveru. Server vytvoří objekt v databázi a upozorní klienta, pokud byl požadavek úspěšný.[10]

Výpis 3.2: požadavek POST

```
curl -i -X POST
--header "Content-Type: application/json"
--data '{ "jmeno": "Andrey",
"prijmeni":"Girin"}'
https://PTS/api/1/user
```

- **PUT** a **PATCH** jsou požadavky, které se používají k aktualizaci informací na serveru. Na základě tohoto požadavku server aktualizuje informace o existujícím objektu a informuje klienta o úspěchu procesu.[10]

Výpis 3.3: požadavek PUT nebo PATCH

```
curl -i -X PUT
--header "Content-Type: application/json"
--data '{"status":"student"}'
https://PTS/api/1/user/185442
</ul>
```

- **DELETE** metoda odstraní objekt z databáze nebo ohlásí chybu, pokud tento objekt nebyl v databázi nalezen.[10]

Výpis 3.4: požadavek DELETE

```
curl -i -X DELETE https://PTS/api/1/user/185442
</ul>
```

4 Použité technologie na straně serveru

Tato kapitola stručně popisuje technologie použité pro implementaci webových serverů, jako jsou programovací jazyk Python, framework Flask, protokol Websokets a MQTT.

4.1 Python

Python je univerzální objektově orientovaný programovací jazyk. Používá dynamickou kontrolu datových typů, což znamená, že typ proměnné je určen její hodnotou a není třeba jej deklarovat. Python je z hlediska učení velmi jednoduchý, a to díky svému krátkému zápisu. Python obsahuje funkce jazyka C a Java, nabízí styl psaní kódu jako v jazyku C a pro objektové programování obsahuje třídy a objekty jako v Javě. Python je víceúčelový jazyk, což znamená, že ho lze používat jak pro objektově orientované, tak pro procedurální programování a v omezené míře i ve funkcionálních paradigmatech, ale to záleží na typu aplikace, pro kterou se hodí nejlépe. Interpret jazyka Python je napsán v několika jazycích, kde „CPython“ je referenční realizace Pythonu napsaná v jazyku C. Existují varianty napsané v jazyce Java, které se nazývají „Jython“, „IronPython“ pro .NET framework a jiné jako „PythonNet“, „PyPy“ atd. Přistup ke zdrojovému kódu C umožňuje zabudovat interpret jazyka Python do jazyka C nebo C++, což rozšiřuje možnosti při tvorbě aplikace. Pomocí překladače se dá „Cython“ převést do čistého C, což má výhodu v rychlosti aplikace. Python podporuje velké množství knihoven. „Pypi.org“ je webový repozitář, kde je možné stáhnout či sdílet vlastní vytvořenou knihovnu. Python aktuálně podporuje verzi 3.8, verze 2.x už není podporovaná. Základní datové typy jazyka Python jsou následující:

- **Booleovské proměnné**
  - **bool(boolean)** jje logický typ, který má pouze dvě hodnoty: true/-false (pravda/nepravda).
- **Numerický typ**
  - **Int (integer)** je celé číslo, jehož velikost omezuje jenom paměť.
  - **Float**je desetinné číslo.
  - **Complex** je komplexní číslo..
- **Seznamy**
  - **List** je pole objektů, které může obsahovat objekty libovolného typu. Každý prvek má svůj vlastní index, začínají od nuly.
  - **Tuple** je pole objektů, ale na rozdíl od listu nelze tento seznam měnit.
- **Řádky**
  - **Str(string)** je řetězec znaků.

- **Slovník**
  - **Dict** je slovník, který ukládá hodnoty do dvojic klíč–hodnota.

Syntaxe jazyka Python je jednoduchá, konec řádku je zároveň konec instrukce a není potřeba používat středník. Vnořený kód je spojen tabulátorem. Přehled cyklů **for** a **while** napsaných v jazyce Python 4.1.

Výpis 4.1: Příklad cyklů „for“ v Python

```
for cislo in range(10):
    print(cislo)
```

Cyklus **for** zapíše postupně 10 čísel v rozsahu od 0 do 9, rozsah nastavuje metoda `range()`.

Výpis 4.2: Příklady cyklů „while“ v Python

```
c = 0
while c < 10:
    print(i)
    c = c + 1
```

Cyklus **while** na rozdíl od for pracuje jinak, a to do okamžiku neplatnosti podmínky. Příklad 4.2 zapíše postupně 10 čísel v rozsahu od 0 do 9.[11]

S pomocí Pythonu je možné automatizovat procesy, vytvářet weby, desktopové aplikace a hry. Existuje mnoho frameworků (softwarových struktur) napsaných v jazyce Python, jako například Django, Flask, Botle nebo CherryPy, pomocí kterých je možné vytvářet webové aplikace.

4.2 Flask

Flask je webový microframework napsáný v jazyce Python. Framework Flask je založen na projektech „Werkzeug“ a „Jinja2“. „Werkzeug“ je knihovna webových aplikací WSGI (Web Server Gateway Interface), která umožňuje zpracovat model komunikace klient-server a „Jinja2“ se zabývá zobrazováním šablon. S pomocí frameworku Flask se dají vytvářet jednoduché i složité aplikace, například boty, weby atd.

Flask umožňuje vývojářům použít vlastní řešení propojení mezi databází a objektově orientovaným programovacím jazykem, který nemá pevně definované ORM(Object-Relational Mapping). Databáze lze rozdělit do dvou velkých skupin: ty, které odpovídají relačnímu modelu (například MySQL, PostgreSQL a SQLite) a ty, které mu neodpovídají. Druhá skupina se často nazývá NoSQL, což znamená, že neimplementuje jazyk SQL (Structured Query Language).[12]

4.2.1 Modely a SQLALCHEMY

Model je jedinečný a konečný sběr dat. Popisuje údržbu a správu klíčových oblastí chráněných dat.

Modely ve Flasku pracují s daty jako jsou zápis a čtení. Vztah mezi nimi je definován jako „One-to-Many“ nebo „Many-to-Many“. Vztahy jsou vyjádřeny pomocí funkce `relationship()`.

Výpis 4.3: Příklad Modelu Uživatel

```
class Uzivatel (db.Model):
    id = db.Column(db.Integer, primary_key=True)
    jmeno = db.Column(db.String(80))
    prijmeni = db.Column(db.String(80))
    email = db.Column(db.String(80), unique=True)
```

`db.Model` je základní třída všech modelů SQLAlchemy. Třída *Uzivatel* je její potomek. V třídě *Uzivatel* z příkladu 4.3 definujeme *id*, *jmeno*, *prijmeni*, *email*. `db.Column` – definuje sloupec v databázi. Pole *id* se používá jako primární klíč v databázi a bude mu přiřazena jedinečná hodnota typu integer. Parametry *jmeno*, *prijmeni* jsou definovány jako řetězce (v databázi je zapsán jako **VARCHAR**). Jejich maximální délka je 80 znaků, zároveň *email*bude mít řetězec stejné délky, ale s parametrem **unique=True**, který toto pole určuje jako unikátní a další stejné pole v databázi nesmí být.

Flask nemá pevně definované ORM (Object-Relational Mapping), aby modely měly možnost pracovat s databází. Flask používá knihovnu **SQLAlchemy**. Daná knihovna převede objekty vytvořené z třídy *Uzivatel* do řádků v databázových tabulkách. Zápis do databáze se řídí následujícím algoritmem 4.5.

Výpis 4.4: Zapiš do databázi

```
>>> from Priklad import Uzivatel
>>> i = Uzivatel('Ivan','Ivanov','i@g.com')
>>> db.session.add(i)
>>> db.session.commit()
```

Na začátku výše uvedeného příkladu je objekt *Uzivatel* (výpis 4.3). Model je přidán do relace pomocí příkazu `session.add()`, ale ještě není vložen do databáze. K potvrzení relace a zadání modelu do databáze je třeba použít příkaz `session.commit()`. Model v databázi je uložen v tabulkách 4.1. Pokud je potřeba získat data z databáze, SQLAlchemy poskytuje atribut `query`. Databáze mohou obsahovat velké množství

Tab. 4.1: Data v databázi SQL

id	jmeno	prijmeni	email
1	Tom	Strom	tomstrom@priklad.com
2	Ivan	Ivanov	i@g.com

dat. Pro zjednodušení práce s databází existují filtry, které umožní načíst požadovaná data. Pro získání správných dat v modelu musí byt nakonfigurována funkce `__repr__(self)`, která bude vracet nastavenou hodnotu. [12] [13]

Výpis 4.5: Výpis z databázi

```
>>> Uzivatel.query.get(2)
<Uzivatel i'Ivan'>
>>>ivan = Uzivatel.query.filter_by(email='i@g.com').first
   ()
>>> ivan.id
2
```

4.2.2 Cesty (Routes) a Pohledy (Views)

Cesty (Routes) jsou implementovány ve tvaru dekoratéru (to je funkce, v níž je zabalena jiná funkce umožňující rozšířit její funkčnost). Ve Flasku pomocí dekoratéru `@app.route()` se funkce navazuje na URL (Uniform Resource Locator), který zobrazuje statické a dynamické cesty. Výpis 4.6 zobrazuje statickou a výpis 4.8 zobrazuje dynamickou cestu.

Výpis 4.6: Přiklad statistické cesty

```
@app.route('/web')
def hello():
    return_'Vraci web'
```

Statická cesta je vhodná na zobrazovaní stránek, kde je třeba pevná cesta.

Výpis 4.7: Přiklad dynamické cesty

```
@app.route('/web/<int:data_id>')
def ukazat_data(data_id):
    return 'Data %d' % data_id.
```

Pomocí dynamické cesty je možné zobrazit data, například vytvořit stránku profilu uživatelů. Cesty zlepšují orientaci uživatelů a usnadňují práci v aplikaci.

Pohledy (Views) jsou implementovány ve tvaru funkce, která se nachází pod dekoratérem `@route`. Funkce **Views** se zobrazí na adrese nastavené na cestě. Pohledy spojují model a šablonu. To znamená, že data modelu lze zobrazit v šabloně a je možné také zadávat požadavky a pracovat s cookies. Pomocí metody „POST“ je možné přidat data do databáze. Pomocí „GET“ se dají data vyžádat. Tyto metody umožňují rozšířit možnosti aplikace.

Výpis 4.8: Zobrazeni všech uživatelů s databázi

```
@app.route('/uzivately/')
def ukazatUzivatelu (page):
    uzivately = Uzivatel.query.all()
    return render_template('uzivately.html',
                           uzivately=uzivately
                           )
```

Funkce z příkladu 4.8 nahrají data uživatelů z modelu pomocí `query.all()`. Metoda `render_template` je zobrazí v šabloně `uzivatel.html`. [12]

4.2.3 Šablony (Tampletes)

Šablony jsou soubory, které obsahují statická data a zástupy pro vstup dynamických dat. Pomocí šablony se vytvoří finální verze dokumentu, s kterou uživatel může komunikovat. Flask používá „Jinja2“ k vykreslování šablon formát HTML (HyperText Markup Languag), XML(eXtensible Markup Language) které pak vracejí uživatelům přes protokol HTTP (HyperText Transfer Protocol). Jinja používá pro úpravu šablony následující oddělovače:

- `{{ ... }}` výrazy pro tisk na výstupu šablony.
- `{% ... %}` používá se pro řídící struktury jako „for“ a „if“.
- `{# ... #}` komentáře, které nejsou součástí výstupu šablony.

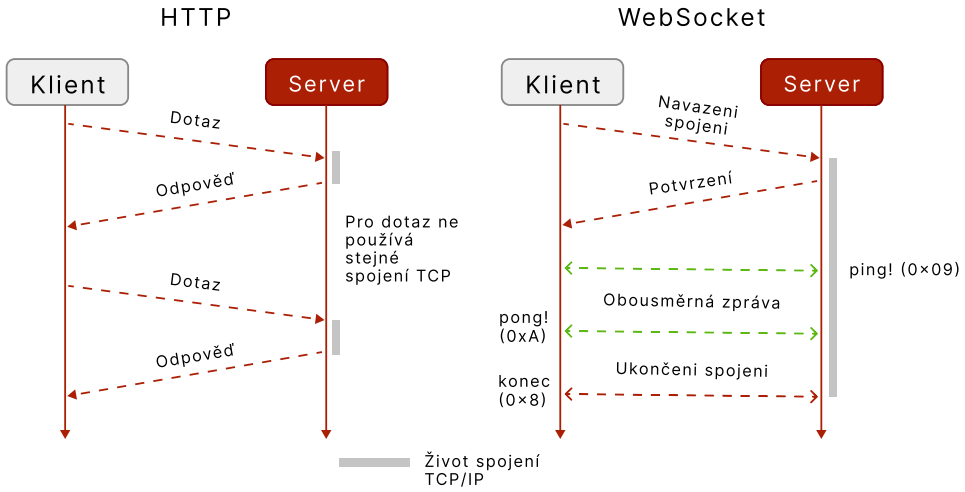
Výpis 4.9: Přiklad cyklusu v šablonovacím jazyku Jinja

```
<h1>Cyklus,ktere vypíše seznam uživatelů</h1>
    <ul>
        {% for uzivatel in uzivately %}
         <li>{{ uzivatel}}</li>
        {% endfor %}

         {# Komentář #}
    </ul>
```

4.3 Websockets

WebSocket je komunikační protokol, který zajišťuje komunikaci mezi klientem a serverem v reálném čase. Protokol je podporován HTML verze 5. Technologie je navržena tak, aby zlepšila standard HTTP. Standard HTTP pracuje na základě *dotaz - odpověď*. WebSocket zůstává pořad otevřený, a proto může přenášet data v reálném čase. Rozdíl mezi HTTP a WebSockets je znázorněn na obrázku 4.1.



Obr. 4.1: Rozdíl HTTP a WebSocket

Výhody oproti HTTP jsou následující::

- Protokol je obousměrný; to znamená, že klient a server si mohou navzájem poslat zprávu, když klient volá HTTP požadavek a server zpracuje odpověď.
- Klient a server spolu mohou komunikovat současně.
- Výměna probíhá přes jedno spojeni TCP (Transmission Control Protocol).

Protokol WebSocket má i své nevýhody, například špatnou kompatibilitu s prohlížečem. Existují různé knihovny pro jejich odstranění, například Socket.IO, která byla použita i v tomto projektu. Knihovna Socket.IO používá k poskytování rozhraní protokol WebSocket. Výhody „Socket.IO“ jsou následující:

- Používá události na straně klienta, jako jsou *Connect*, *Message*, *Disconnect*, *Ping* a *Reconnect*. V případě chyby se aktualizuje připojení.
- Pokud klient nepodporuje webový soket, pak se „Socket.io“ může přepnout na `POST` a `GET`.
- Může vysílat na více soketů současně.

4.4 MQTT (Message Queue Telemetry Transport)

MQTT je protokol pro výměnu zpráv mezi zařízeními a pracuje na principu *Publish/Subscribe* publikovavat/odebírat. Protokol se používá pro zařízení s nízkým výkonem nebo v sítích s omezenou šířkou pásma, a proto se používá I v systémech IoT (Internet of Things). Protokol se skládá z *„MQTT Broker“* a *„MQTT Client“*.

- Broker je centrální úložiště, které sbírá všechny zprávy od klienta. Klient může být vydavatel i odběratel (může posílat nebo přijímat zprávy).
- Klient může být vydavatel i odběratel (může posílat nebo přijímat zprávy).

Popis zpráv mezi Klientem a Brokerem je následující: vydavatel odešle zprávy, které se týkají konkrétního tématu (topic). Odběratel může tato data obdržet, pokud je přihlášen k tomuto tématu. Topic má hierarchickou strukturu typu „strom“ a skládá se z jedné nebo více úrovní, které jsou odděleny symbolem „/“:

```
Dum/PrvniPatro/pokoj_1/temperatura
```

Pro získání dat z několika **topiců** se použijí zástupné znaky. Znak „+“ zde nahrazuje jednu vrstvu. Díky tomu je možné získat data ze všech místností na prvním patře:

```
Dum/PrvniPatro/+/temperatura
```

Pomocí zástupného znaku c e možné získat všechna data ze zařízení, která jsou napatře.[14]

```
Dum/PrvniPatro/#
```

4.4.1 Struktura zprávy

Zpráva MQTT se skládá z několika částí v následujícím pořadí: pevná hlavička, proměnná hlavička, zatížení. Proměnná hlavička a zatížení nemusí obsahovat všechny řídicí pakety, v tabulce 4.3 sou uvedeny příklady.

Pevnou hlavičku („Fixed Header“) obsahuje každá zpráva. Struktura pevné hlavičky je zobrazena na obrázku 4.2

- **Typ řídícího paketu:** jedná se o zprávy jako CONNECT, SUBSCRIBE, PUBLISH.
- **Příznaky specifické pro každý typ řídícího paketu MQTT:** 4 bity přidělené pro příznaky obr.4.4.
- **Zbývající délka:** délka zprávy, proměnná záhlaví a data.

DUP - jedná se o duplikát. Pokud je hodnota 0, zpráva byla doručena poprvé, pokud je hodnota příznaku 1, tato zpráva byla odeslána opakovaně. (Používá se v PUBLISH, SUBSCRIBE, UNSUBSCRIBE, PUBREL.)

QoS - tento příznak určuje úroveň kvality služeb..

Bit	7	6	5	4	3	2	1	0
byte 1	Typ řídícího paketu				Příznaky specifické pro každý typ řídícího paketu MQTT			
byte 2	Zbývající délka							

Obr. 4.2: Pevná hlavička

3	2	1	0
DUP	Qos	Qos	Retain

Obr. 4.3: Bity přidělené pro příznaky

- **0** - Vydavatel zašle na Broker jednu zprávu a nečeká na potvrzení jejího přijetí.
- **1** - Tato úroveň zajišťuje přesné doručení zprávy na Broker, ale existuje možnost, že zpráv může být více.
- **2**- Je zaručeno doručení zprávy na Broker a je vyloučena možná duplikace odeslaných zpráv.

RETAIN - Pokud je nastaven tento příznak, Broker tuto zprávu uloží. Když se klient přihlásí k odběru tohoto tématu, Broker odešle tuto zprávu odběratelům. Používá se ve zprávách typu `PUBLISH`.

Proměnná hlavička (Variable Header) bsahuje některé typy řídicích paketů MQTT. Obsah proměnné hlavičky závisí na typu paketu. Obvykle obsahuje 2 bity MSB a LSB identifikátory daného paketu. daného paketu. Identifikátory obsahují pakety PUBLISH (kde je QoS > 0), PUBACK, PUBREC, PUBREL, PUBCOMP, SUBSCRIBE, SUBACK, UNSUBSCRIBE, UNSUBACK.

Bit	7	6	5	4	3	2	1	0
byte 1	MSB identifikátor paketu							
byte 2	LSB identifikátor paketu							

Obr. 4.4: Bity přidělené pro příznaky

Zatížení (Payload) obsahuje data přenášená ve zprávách MQTT. Toto pole není vždy zahrnuto do zprávy a obsah zatížení je různý, a to v závislosti na typu řídicího paketu (viz tabulka 4.3). Obsah zatížení v případě typu paketu CONNECT

je identifikátor klienta, jméno, heslo. V případě, že je typem paketu PUBLISH, data obsahují pouze aplikační zprávu. [14]

Tab. 4.2: Pevná hlavička ,Proměnná hlavička, Zatížení řídících paketu.

Řídící paket	**Hodnota**	**Pevná hlavička**	**Proměnná hlavička**	**Zatížení**
CONNECT	0001 (1)	+	-	+
CONNACK	0010 (2)	+	-	-
PUBLISH	0011 (3)	+	+ (QoS >0)	Volitelné
PUBACK	0100 (4)	+	+	-
PUBREC	0101 (5)	+	+	-
PUBREL	0110 (6)	+	+	-
PUBCOMP	0111 (7)	+	+	-
SUBSCRIBE	1000 (8)	+	+	+
SUBACK	1001 (9)	+	+	+
UNSUBSCRIBE	1010 (10)	+	+	+
UNSUBACK	1011 (11)	+	-	-
PINGREQ	1100 (12)	+	-	-
PINGRESP	1101 (13)	+	-	-
DISCONNECT	1110 (14)	+	-	-
Rezervováno	1111 (15)	Zakázáno	Zakázáno	Zakázáno

Tab. 4.3: Popis všech řidičích paketu MQTT.

Řídící paket	**Popis řidičích paketu**	**Komunikace**
CONNECT	Klient požaduje připojení k serveru	K-S
CONNACK	Klient požaduje připojení k serveru	S-K
PUBLISH	Probíhá publikace zprávy	S–K,K–S
PUBACK	Potvrzení publikace zprávy	S–K, K–S
PUBREC	Publikace zprávy je přijata	S–K, K–S
PUBREL	Povolení odstranit zprávu	S–K, K–S
PUBCOMP	Publikace zprávy je ukončena	S–K, K–S
SUBSCRIBE	Klient žádá server o předplatné	K-S
SUBACK	Server přijal od klienta žádost o předplatné	S-K
UNSUBSCRIBE	Klient posíla na server žádost o zrušení předplatného	K-S
UNSUBSCRIBE	Server přijal žádost klienta o zrušení předplatného	S-K
UNSUBSCRIBE	Klient posílá na server ping dotaz	K-S
PINGRESP	Server odpovídá klientovi na ping dotaz	S-K
DISCONNECT	Zpráva o odpojení klienta ze serveru	S-K

5 Použité technologie na straně klientovi

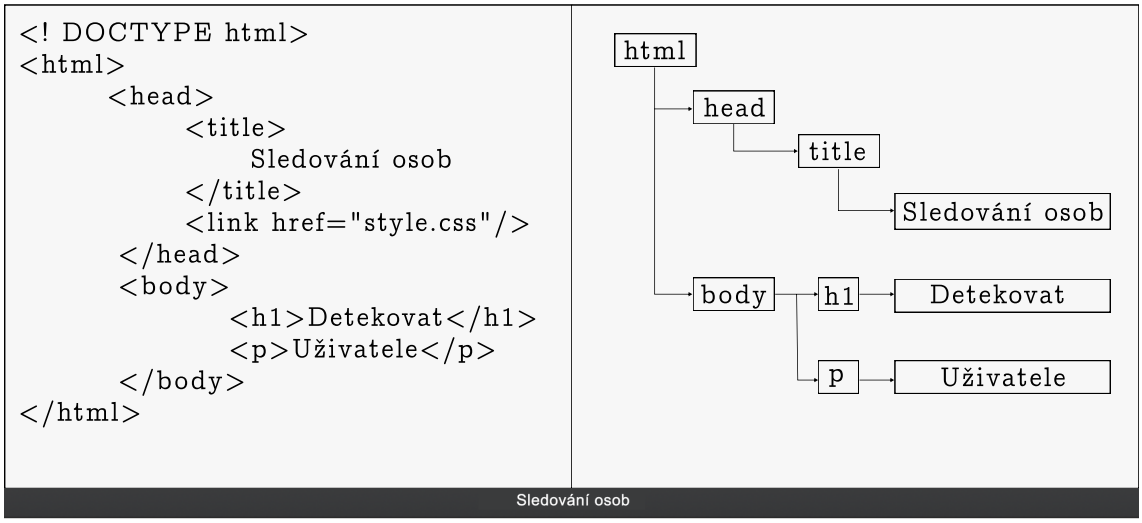
Tato kapitola stručně popisuje použity technologie na klientské straně aplikace takový jako značkovací jazyk HTML(HyperText Markup Language), stylovací jazyk CSS(Cascading Style Sheets) a programovací jazyk Javascript.

5.1 HTML (HyperText Markup Language)

HTML je značkovací jazyk, který lze použít k zobrazení obsahu hypertextového dokumentu ve webovém prohlížeči. Každý HTML dokument od verze „HTML5“ začíná značkou `<!DOCTYPE html>`. HTML je založen na značkách „tag“, s jejichž pomocí se může do dokumentu přidat text, obrázek nebo video, které umožní uživatelů pracovat s webovou stránkou. Například pomocí značky `<h1>` lze vytvořit největší nadpis a pomocí značky `<h6>` nejmenší. Značka `<img>` umožňuje publikovat obrázek. Většina značek v jazyce HTML je párová, což znamená, že obsah je ohraničen dvěma značkami.

Základní značky HTML jsou následující::

- `<html>...<html>` ukazuje webovému prohlížeči, že se jedná o dokument HTML..
- `<head>...<head>` definuje hlavičku dokumentu. Obsahuje značku názvu dokumentu a značky pro vyhledávače (cesty k CSS nebo JS souborům).
- `<body>...<body>` definuje viditelnou část dokumentu.



Obr. 5.1: Příklad DOM, a finální zobrazeni v prohlížeče

DOM (Document Object Model) je reprezentace objektu původního HTML dokumentu a první krok při jeho vykreslení v prohlížeči. Všechno v HTML, dokonce i komentáře, je součástí DOM. Díky DOM může JavaScript komunikovat se stránkou, měnit její obsah, strukturu a styly. Díky DOM můžete sledovat klientské události (viz obrázek 5.1). DOM je založen na HTML a nezohledňuje styly CSS.

5.2 CSS (Cascading Style Sheets)

CSS je stylovací jazyk, který definuje způsob zobrazení HTML dokumentů vytvořených pomocí značkovacího jazyka HTML. CSS pracuje s písmy, znaky a barvami pozadí, s okraji, řetězci, výškou a šířkou prvků. Styl se skládá ze selektoru a části v závorkách, která se nazývá blok deklarací. Selektor ukazuje na objekt HTML, na který se pravidlo použije. Blok deklarací se skládá ze dvou částí: názvu vlastnosti a hodnoty; vzájemně jsou oddělené dvojtečkou. Selektor například ukazuje prohlížeči objekt `<h1>`, blok deklarace (umístěný ve složených závorkách `{...}`) uvádí příkazy formátování, v tomto případě vlastnost `font-size` nastaví velikost písma, hodnota `1rem` ukazuje velikost textu v objektu `<body>...</body>` výpis A.4. Aby pravidla definovaná v CSS souboru platila i v HTML dokumentu, je potřeba odkázat na CSS soubor v hlavičce HTML dokumentu `<head>...</head>` pomocí nepárové značky `<link>`, uvnitř které bude definovaná cesta k souboru.

```
<! DOCTYPE html>
<html>
      <head>
           <title>
               Sledování osob
           </title>
           <link href="style.css"/>
       </head>
       <body>
              <h1>Detekovat</h1>
              <p>Uživatele</p>
       </body>
</html>
```

```
body {
      font-size: 1rem;
      border-left: 6px  solid #ff5200;
      background-color: lightgrey;
      font-family: Helvetica;
}
h1 {
      color: #161b09;
      margin-left: 0.5rem;
}
p {
     color: #161b09;
     margin-left: 0.5rem;
}
```

Obr. 5.2: Použití stylu CSS k HTML souboru a zobrazeni v prohlížeči

5.3 Bootstrap

Bootstrap je otevřený a bezplatný HTML, CSS a JS framework pro rychlé zavedení responzivních návrhů webových stránek a webových aplikací. Framework představuje sadu CSS a JS souborů. Pro jejich používaní stačí připojit soubor ke stránce. Knihovna bude připojena pro použití dostupnými nástroji Bootstrapu: mřížkovým systémem, třídami a dalšími komponenty. Bootstrap tvoří tyto části:

- mřížkový systém;
- třídy pro styling textu, obrázků, tabulek a jiného obsahu;
- součásti určené k vytváření tlačítek, různých formulářů na stránce, horizontálních a vertikálních navigačních nabídek, posuvníků, rozevíracích seznamů, akordeonů, modálních oken, popisků a dalších prvků rozhraní;
- řídy pro řešení pomocných problémů (zarovnání textu, skrytí nebo zobrazení prvku, nastavení barvy a pozadí prvku, nastavení okraje a odsazení výplně atd.)

5.4 JavaScript

JavaScript je multiplatformový, objektově orientovaný skriptovací jazyk zaměřený na události. JavaScript se používá na klientské straně webových aplikací. U programů klient-server, ve kterých je klientem prohlížeč, může JavaScript následující:

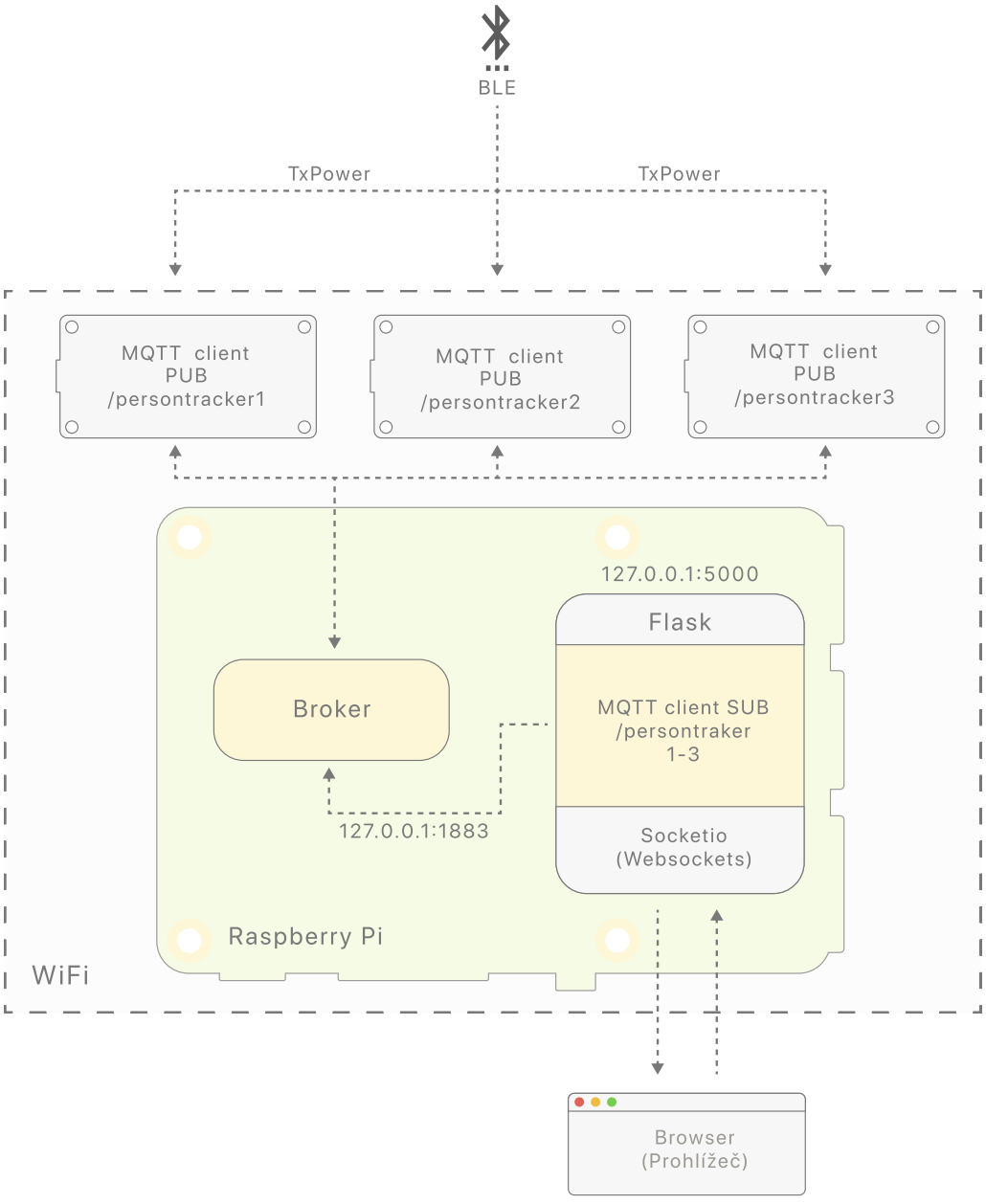
- V prohlížeči přidat nový HTML kód na stránku nebo upravit existující obsah a styly.
- Reagovat na akce uživatele, kliknutí myši, pohyb ukazatele nebo stisknutí kláves.
- Odesílat síťové požadavky na vzdálené servery, stáhnout a nahrávat soubory (technologie „AJAX“ a „COMET“).
- Přijmout a nastavovat cookies, zeptat se návštěvníka, zobrazovat zprávy..
- Ukládat data na straně klienta.

JavaScript pracuje se stránkou pomocí DOM, což umožňuje provádět cokoli s prvkem HTML a jeho obsahem Integrace JavaScriptu do HTML stránky probíhá vložením skriptu mezi dvojici značek `<script>` a `</script>`, a to z externího souboru určeného atributem `src` značky `<script>` nebo pomocí obslužné rutiny události nastavené jako hodnota atributu HTML, jako je `onclick` nebo `onmouseover`.

6 Realizace projektu

Tato kapitola popisuje technickou realizaci projektu, konkrétně interakci hardwaru, programování mikrokontrolérů a jejich umístění v místnosti.

6.1 Princip fungování a struktura projektu



Obr. 6.1: Architektura projektu

Klient „MQTT pub“ publikuje data čtená mikrokontrolérem z prostředí a publikuje je jako „**topic**“ na brokerovi. „**Topic**“ jsou znaky kódované „UTF8“. Délka

tématu nesmí přesáhnout 32 767 znaků. Projektová data jsou čtena mikrokontrolérem ESP32.

Zprostředkovatel (Broker) je nejdůležitější součástí systému „vydavatel-odběratel“. Zprostředkovatel je odpovědný za přijímání všech zpráv, jejich filtrování, rozhodování o tom, kdo se o tyto zprávy zajímá, a nakonec za zasílání zpráv všem klientům předplatitele. Mezi serverové implementace brokera existuje mnoho možností, ale tento projekt využívá „**Mosquitto**“. Broker pracuje paralelně s webovým serverem na jednom zařízení a je k dispozici na portu **1883**.

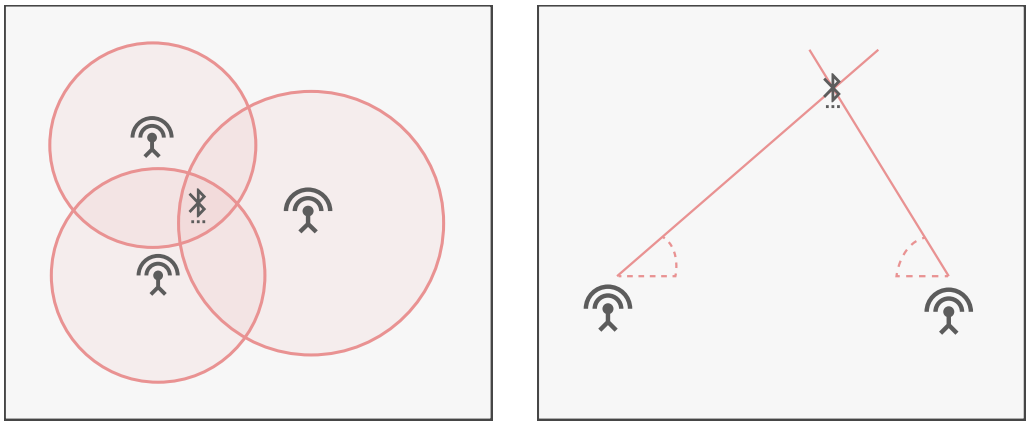
Webový server je ústředním prvkem, který se přihlásí k odběru tématu publikovaného brokerem. Přijatá data z tématu zpracovává a přiřazuje uživatel. Zpracovaná data jsou v reálném čase přenášena do prohlížeče pomocí protokolu „Websoket“. Webový server je implementován na mikroframeworku Flask napsaném v jazyce Python. Prohlížeč je grafická část systému. Bere asynchronní data odeslaná webovým serverem a zpracovává je pomocí JS. Na základě zpracovaných dat algoritmus mění parametry prvku v CSS a poloha osoby je zobrazena na online mapě.

6.2 Určení souřadnic majáku BLE

Pro detekce mobilního uzlu v místnosti je nutné do ní umístit BLE majáky. Čím větší bude jejich počet a čím menší budou vzdálenosti mezi nimi, tím přesněji můžeme určit souřadnice zařízení.Majáky je nutné uspořádat tak, aby rovnoměrně pokrývaly celou plochu místnosti s výpočtem vzdáleností majáků od sebe navzájem. Do rohu místnosti lze položit počátek souřadnic.

Rozlišují se dvě metody relativní detekce mobilního uzlu: trilaterace a triangulace (rozdíl mezi těmito metodami ukazuje obrázek **??**). Během trilaterace se určuje vzdálenost mezi majákem a zařízením, které lze umístit do průsečíku kruhů s poloměrem rovným vzdálenosti k majáku (implementace vyžaduje tři podpůrné uzly). Při triangulaci se souřadnice zařízení počítají na základě souřadnic referenčních objektů a úhlů z každého z nich. Kde bude zařízení v průsečíku, tam k určení postačují dva referenční uzly. TX Power je RSSI (Received Signal Strength Indicator), který ukazuje sílu signálu majáku, jmenovitě sílu signálu ve vzdálenosti jednoho metru od majáku. K určení polohy v místnosti je třeba znát souřadnice majáků v prostoru a vzdálenost k nim, za kterou je zodpovědný RSSI. Na základě síly přijímaného signálu je možné najít zařízení v prostoru. Teoreticky stačí umístit několik zařízení v různých úhlech v prostoru a vypočítat úroveň signálu, ve skutečnosti však parametr RSSI náhodně mění své hodnoty, a to z těchto příčin:

- Kvůli přítomnosti velkých stínících objektů ve směru zařízení od majáku (například i osoba může způsobit rušení).).



Obr. 6.2: Rozdíl mezi trilaterace a triangulace

- Z důvodu velké akumulace majáků a povrchů materiálů odrážejících rádiový signál.
- Směr a směrová charakteristika záření nebo příjmu majáku / zařízení.

Pomocí trilaterace je možné vyloučit detekci mimo oblasti a získat přesnost kolem 3-4 metrů. Pro přesnější určení je nutné matematické zpracování získaných dat a jejich korekce, protože při deseti po sobě jdoucích měřeních mohou být získána odlišná RSSI data.

6.3 ESP

Pro definovani polohy osoby v místnosti, potreba získat data ze uzivatelskoho bluetooth zařízení . Data jsou čtena pomocí mikrokontroléru ESP32, ale aby zařízení mohlo číst data, musí být naprogramováno. Pro programování mikrokontroléru se používá systém EPSHOME.

6.3.1 ESPHOME

Systém ESPHOME je tvořen sadou nástrojů a knihoven, které generují mikroprogramové vybavení pro mikrokontrolér ESP32 z konfiguračního souboru vytvořeného uživatelem ve formátu YAMAL. Tento přístup umožňuje rychle a efektivně nakonfigurovat firmware pro mikrokontrolér ESP32, protože systém podporuje velké množství modulů a uživatel nepotřebuje hluboké znalosti jazyka C.

6.3.2 DATA

Mikrokontrolér ESP32 podporuje technologii Bluetooth Low Energy. Díky tomu je mikrokontrolér schopen přijímat a odesílat reklamní pakety BLE. Na základě re-

klamních paketů mikrokontrolér přijímá data ze zařízení, aniž by se k nim připojoval. Zařízení poskytují následující data: typ adresy (statický nebo náhodný), adresu zařízení a úroveň jejich signálu (RSSI).

Výpis 6.1: Bluetooth Low Energy Scanner

```
esp32_ble_tracker:        # Skenuje zařízení BLE
text_sensor:              # překládá v formát String
  - platform: ble_scanner # Skenuje zařízení
    name: "BLE Devices Scanner"
```

Na základě těchto dat mikroprogramových vybavení mikrokontrolér skenuje reklamní kanály v přísně omezeném pořadí: 37 (2 402 MHz), 38 (2 426 MHz) a 39 (2 480 MHz). Dostupná zařízení Senzor BLE monitoruje a převádí do textového řetězce ve formátu JSON. Řetězec je přenášen pomocí protokolu MQTT a publikován na Brokeru, kde se následně objeví na serveru.

Výpis 6.2: přijata data v formátu JSON

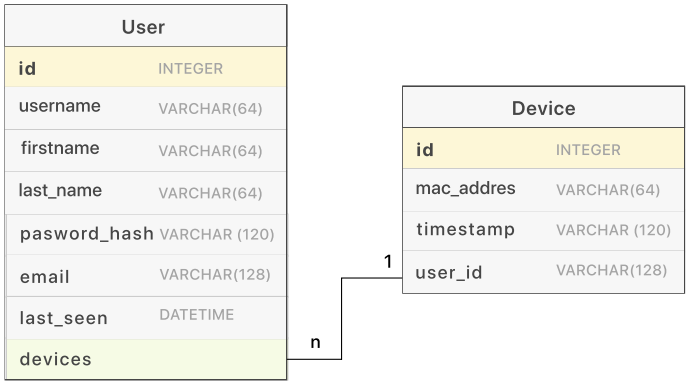
```
{
    "timestamp":1578254525,
    "address":"CB:50:E3:A8:2D:9C",
    "rssi":"-80",
    "name":"xpgirin00"
}
```

7 Realizace web serveru

Serverová část aplikace napsané v mikroframeworku Flask pro jazyk Python. Server je zodpovědný za práci s databází, zpracování požadavků, vytváření formulářů pro vazbu, vytváření adres URL, přijímání dat od Brokeru a jejich odesílání do prohlížeče. Aplikace lze rozdělit do několika částí, konkrétně do autentifikace, hlavního menu a uživatelského profilu.

7.1 Models

Důležitou součástí programu je zpracování a ukládání přijatých dat. Projekt používá databázi Postgresql k ukládání dat. Modely označují, jaká data budou uložena a vztah mezi nimi. V projektu pro ukládaní dat uživatelů a informace o jejího zařízeni použity následující modely:



Obr. 7.1: Aktuálně web-server.

7.1.1 User (uživatel)

Ukládá data o registrovaných uživatelích a zařízeních, která k nim patří. Pole **ID** typu **Integer** je individuální identifikátor uživatele. Pole **username** a **email** typu **String** je unikátní. V poli **username** je uložena uživatelské jméno a v **email** e-mailová adresa uživatele.**first_name** a **lastname_name** typu **String** uloženo jméno a příjmení uživatele. Pole **password_hash** typu **String** v tomto pole je heslo, které uloženo v podobě hashu. Pole last_seen typu Datetime zobrazuje poslední aktivitu uživatele. Pole devices ma vztah mnoha souvisejících s objektem.

7.1.2 Device

Model zařízení ukládá informace o zařízeních, která si uživatel přidal k sobě. Na základě zde uložených dat je zařízení vázáno svou vyskakovací adresou. Model se skládá z následujících polí: **id**, **Mac_adresa**, **user_id**. Pole **id** typu **integer** je jedinečným indikátorem zařízení v tabulce. Pole **Mac_address** typu **VARCHAR** ukládá data o mac adrese zařízení jako text. Pole **User_id** typu **VARCHAR** vrací informace o uživateli, kterému zařízení patří.

7.2 Autorizace

Aby byla uživateli přidělena přístupová práva ke zdroji, musí se uživatel v systému autorizovat. Díky autorizaci se uživatel stává součástí systému a také má představu o tom, kdo k němu má přístup. Autorizace je implementovaná pomocí knihovny „flask-login“. Když uživatel spustí aplikaci v prohlížeči, nejprve se mu zobrazí autorizační okno. Existují tři možnosti, které uživatel má:

- Autorizace (pokud je již uživatel v systému zaregistrován),
- Registrace do systému,
- Reset hesla (pokud je uživatel zaregistrován, ale nepamatuje si své heslo).

Obr. 7.2: Použití stylu CSS k HTML souboru a zobrazeni v prohlížeči

7.2.1 Autorizace

Oprávnění uživatele je v systému povoleno pomocí jeho uživatelského jména nebo e-mailu. Během autorizace probíhá na serveru následující proces: uživatel přenáší svá data do formulářů implementovaných pomocí knihovny „Flask-WTF“. Data z

formuláře jsou porovnána s hodnotami uloženými v databázi a pokud se shodují, uživatel je přesměrován na hlavní stránku aplikace pomocí metody `redirect`.

7.2.2 Registrace

Registrace v aplikaci je vyžadována pro přístup do systému, takže je aplikace chráněna metodou `@login_required`. Na registrační stránce uživatel prochází identifikací, ve které předem přiřadí své osobní údaje, jméno a příjmení. Registrační pole se skládá z následujících polí: „email“, `username`, `first_name`, `last_name`, `password`, `password_repeat` (pro zopakování hesla). Server porovná přijatá data s daty v databázi a pokud již existuje pole pro stejný e-mail nebo uživatelské jméno, server o tom uživatele informuje. Když uživatel potvrdí jedinečnost svých dat, zapíší se do databáze. Z důvodu zabezpečení osobních údajů uživatelů jsou v databázi uložena jako „hash“. Poté, co uživatel přidá svá data do databáze, bude přesměrován na stránku „login“ pomocí metody `redirect` a bude mu zaslán e-mail k potvrzení registrace.

7.2.3 Reset hesla

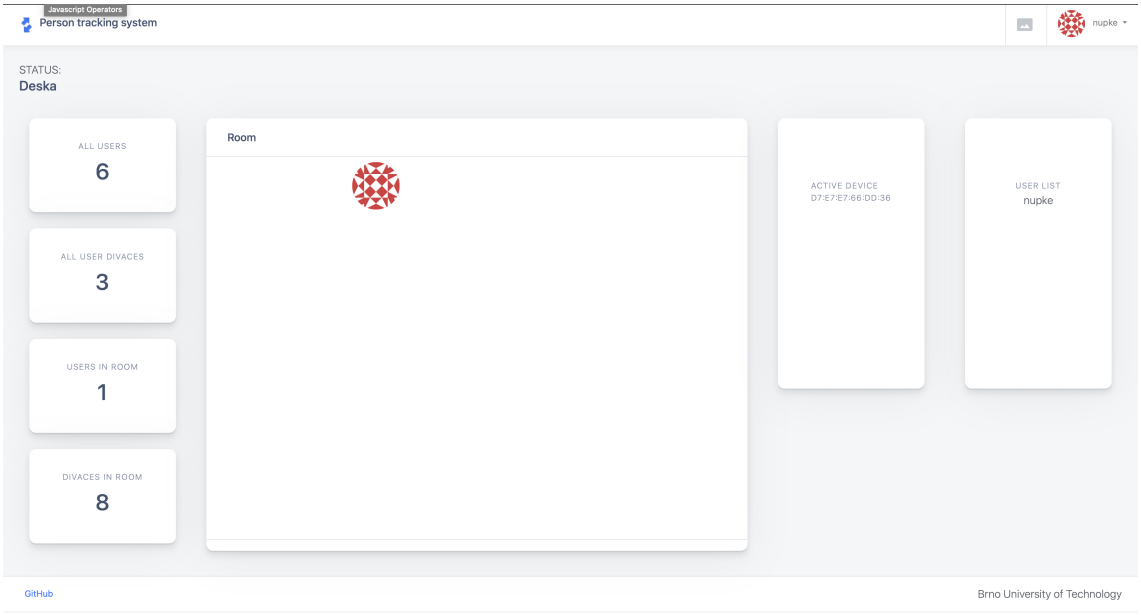
V případě, že je uživatel zaregistrován v systému, ale nepamatuje si heslo, aplikace poskytuje funkci resetování hesla. Pro resetování hesla musí uživatel do pole napsat svou e-mailovou adresu a server odešle data na jeho poštu.

7.3 Hlavní stránka (index) a Uživatelský profil

Tato oblast obsahuje rozhraní aplikace. V levě části aplikace jsou statistické udaje jako: počet registrovaných uživatelů a zařízení, jakož i počet návštěv v místnosti. Tyto statistiky fungují následovně. Server bude odesílat data z databáze a Javascript zpracovává a publikuje je.

K implementaci interaktivní mapy, kde bude zobrazeno aktuální pozice uživatele, je nutné porovnat data z databáze a mikrokontrolérů. Poskytování dat ze serveru o uživatelích a jejich zařízeních je implementováno pomocí API, kde jsou data přenášena ve formátu JSON.

Server přijímá data od brokerovi a publikuje je asynchronně v prohlížeči pomocí knihovny „Socket.IO“. V prohlížeči jsou data zpracovávána pomocí jazyka JS. Publikovaná data jsou řazena podle topiců. Poté jsou tříděná data porovnána s daty z databáze. Pokud existuje shoda mezi daty ze serveru a mikrokontroléru, pak je uživatelskému zařízení přiřazena úroveň signálu přijímaná z mikrokontroléru. Přiřazení dat ze zařízení k uživatelům se provádí pomocí funkce 7.1

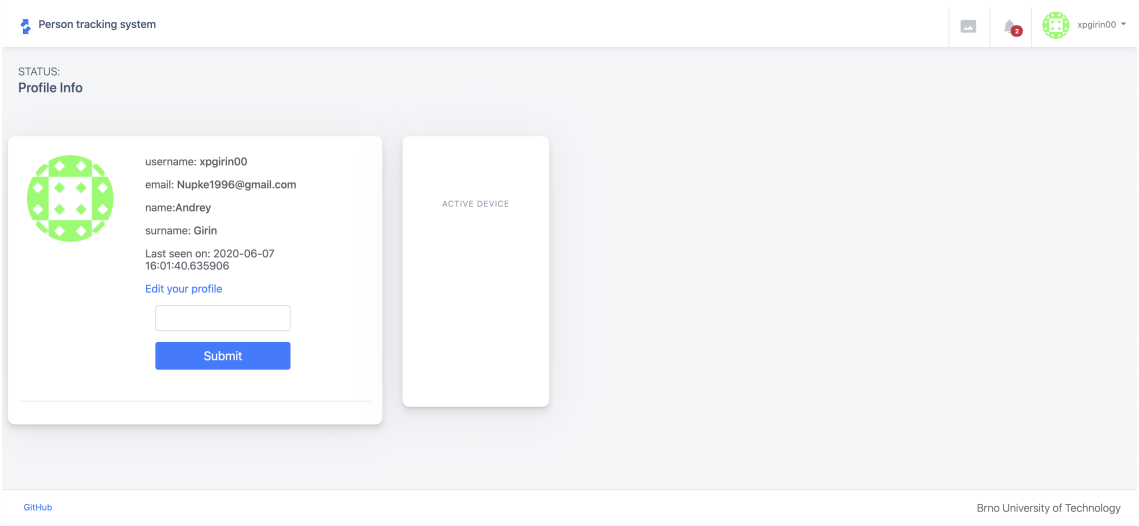


Obr. 7.3: Hlavni stránka aplikace

Výpis 7.1: Přiřazení dat ze zařízení k uživatelům

```
function connectTopic(drone, address, rssi) {
    for(let i=0; i<users.length;i++){
        for(let y=0; y<users[i].devices.length;y++){
            if(users[i].devices[y].mac_addres==address){
               users[i].devices[y].position[drone]=rssi
            }
          }
        }
    }
```

Pro interakci uživatele se systémem musí poskytnout údaje o svém zařízení, konkrétně MAC adresu zařízeni. Uživatel může mít několik registrovaných zařízení. V systému však lze zařízení zaregistrovat pouze pro jednoho uživatele. Zaregistrovat zařízení lze v profilu uživatele



Obr. 7.4: Uživatelský profil

Závěr

Cílem této bakalářské práce bylo vytvořit systém pro sledování osoby v místnosti a její zobrazení na virtuální mapě. Všechny kroky zadání byly splněny, aplikace sleduje aktuální polohu osobu v místnosti a tuto osobu identifikuje. Toto řešení může být použito v systému chytré domácnosti pro interakci s jeho elementy na základě polohy uživatele v místnosti, čímž poskytuje jedinečný scénář hlavních osobních preferencí. Tento systém není omezen počtem uživatelů, ale má omezení ve formě fyzického prostoru, ve kterém dochází k detekci.

Vytvořený systém se skládá ze tří částí. Aplikace pro modul ESP32, serverovou aplikaci a webové rozhraní aplikaci. Hlavním úkolem aplikace pro modul bylo hledání zařízení v síti Bluetooth a odeslání přijatých dat na webový server. Aplikace byla vytvořena pomocí modulárního systému esphome, který je srozumitelný díky dobře napsané dokumentaci. Serverová část aplikace musí podporovat protokol MQTT ke shromažďování dat z mikrokontrolérů a protokolu Websockets pro jejich zveřejnění v reálném čase. Pro implementaci serverové aplikace bylo na výběr mezi frameworkem Django a mikroframeworkem Flask, napsaných v jazyce Python. Framework Django nesplnil požadavky kvůli špatné kompatibilitě protokolu MQTT a protokolu Websocets. Pro realizaci projektu byl proto vybrán Flask, který vyhovoval všem požadavkům projektu a měl velké množství zdrojů pro nastavení komunikace mezi MQTT a Websocets. Aplikační rozhraní bylo vytvořeno pomocí šablony a knihovny Jquery pro Javascript.

Tento projekt má v budoucnu šanci na široké využití. V současné verzi aplikace však není implementován systém pro vytvoření interaktivního plánu místnosti a vzdálenou konfiguraci modulu Esp32. Z tohoto důvodu implementace tohoto projektu vyžaduje určitou úroveň znalostí pro konfiguraci systému.

Literatura

[1] Inc. Microchip Technology. Microchip: Bluetooth® low energy connection process, 2019. [Online; navštíveno 22.10.2019]. URL: https://microchipdeveloper.com/wireless:ble-link-layer-packet-types.

[2] E. BAJIC F. CHAXEL a F. MEYER. MEKKI, K. A comparative study of lpwan technologies for large-scale iot deployment, 2017. [Online; navštíveno 28.05.2020]. URL: https://www.sciencedirect.com/science/article/pii/S2405959517302953.

[3] Inc. Bluetooth Special Interest Group. Bluetooth: Bluetooth sig - about us, 2019. [Online; navštíveno 22.10.2019]. URL: https://www.bluetooth.com/about-us/.

[4] Inc. Microchip Technology. Microchip: Bluetooth® low energy connection process, 2019. [Online; navštíveno 22.10.2019]. URL: https://microchipdeveloper.com/wireless:ble-phy-layer.

[5] Inc. Microchip Technology. Microchip: Bluetooth® low energy connection process, 2019. [Online; navštíveno 22.11.2019]. URL: https://microchipdeveloper.com/wireless:ble-link-layer-address.

[6] Wikipedie. Raspberry pi — wikipedie: Otevřená encyklopedie, 2019. [Online; navštíveno 4. 11. 2019]. URL: https://cs.wikipedia.org/w/index.php?title=Raspberry_Pi&oldid=17874477.

[7] Espressif. Espressif: Esp32 datasheet version 2.1, 2019. [Online; navštíveno 1.11.2019]. URL: https://www.espressif.com/en/products/hardware/modules.

[8] Espressif. Esp32-wroom-32d & esp32-wroom-32u datasheet version 1.9, 2019. [Online; navštíveno 1.11.2019]. URL: https://www.espressif.com/en/products/hardware/modules.

[9] John Deacon. Model-view-controller (mvc) architecture. *Online][Citado em: 10 de março de 2006.] http://www. jdl. co. uk/briefings/MVC. pdf*, 2009.

[10] M Masse. *REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces*. "O'Reilly Media, Inc.", 2011.

[11] Mark Lutz. *Learning python: Powerful object-oriented programming*. "O'Reilly Media, Inc.", 2013.

[12] Miguel Grinberg. *Flask web development: developing web applications with python*. "O'Reilly Media, Inc.", 2018.

[13] Rick Copeland. *Essential sqlalchemy*. "O'Reilly Media, Inc.", 2008.

[14] OASIS. Mqtt 3.1.1, 2014. [Online; navštíveno 5.12.2019]. URL: `http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html`.

[15] Inc. Microchip Technology. Microchip: Bluetooth® low energy connection process, 2019. [Online; navštíveno 22.11.2019]. URL: `https://microchipdeveloper.com/wireless:ble-link-layer-connections`.

[16] RASPBERRY PI FOUNDATION. The raspberry pi bcm2837, 2019. [Online; navštíveno 28.10.2019]. URL: `https://www.raspberrypi.org/documentation/hardware/raspberrypi/bcm2837/README.md`.

[17] Espressif. Esp32-wroom-32d & esp32-wroom-32u datasheet version 1.9, 2019. [Online; navštíveno 1.11.2019]. URL: `https://www.espressif.com/en/products/hardware/modules`.

[18] Guilherme MB Oliveira, Danielly CM Costa, Ricardo JBVM Cavalcanti, Josiel PP Oliveira, Diego RC Silva, Marcelo B Nogueira, and Marconi C Rodrigues. Comparison between mqtt and websocket protocols for iot applications using esp8266. In *2018 Workshop on Metrology for Industry 4.0 and IoT*, pages 236–241. IEEE, 2018.

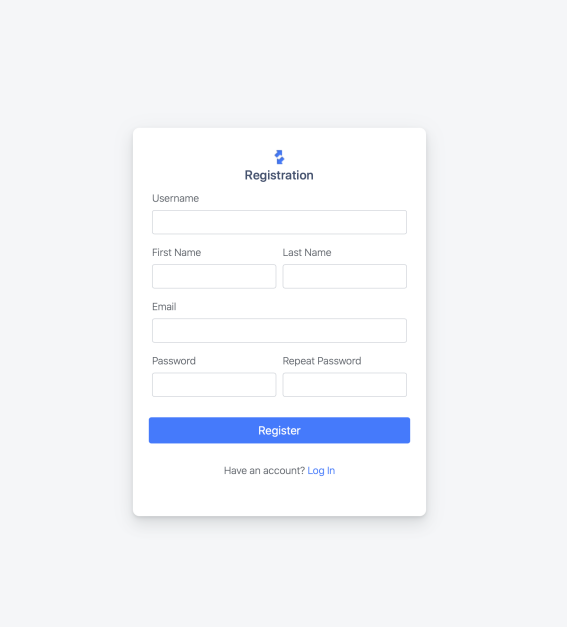
Seznam symbolů, veličin a zkratek

ACK	Acknowledgment
AdvA	Advertiser device address
ARM	Advanced RISC Machine
BLE	Bluetooth nízké spotřeby energie – Bluetooth Low Energy
CRC	Cyclic redundancy check
CSS	Cascading Style Sheets
DOM	Document Object Model
GUI	Graphical User Interface
GHz	Giga Hertz
HTML	HyperText Markup Languag
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
IDE	Integrated Development Environment
IoT	Internet věcí – Internet of Things
JS	JavaScript.
JSON	JavaScript Object Notation
kHz	Kilohertz
LoRaWAN	Low Power Wide Area Network
LPWA	Long Range Wide Area Network
M2M	Machine-to-Machine
MAC	Media Access Control
MHz	Mega Hertz
MQTT	Message Queuing Telemetry Transport
MVC	Model View Controller
ORM	Object-relational mapping
QoS	Quality of Service
OUI	Organizationally Unique Identifier
PAN	Personal Area Network
PDU	Protocol Data Unit
QoS	Quality of Service
REST	Representational State Transfer
RSSI	Received Signal Strength Indicator
SQL	Structured Query Language
TCP	Transmission Control Protocol
UHF	Ultra High Frequency
UUID	Universally unique identifier
URL	Uniform Resource Locator

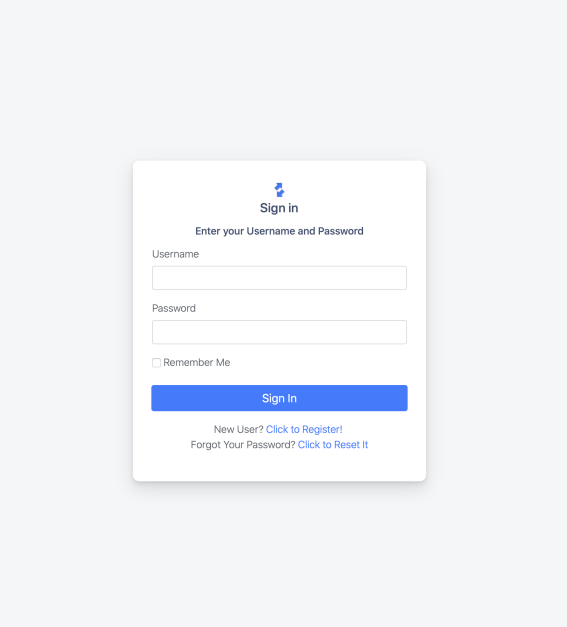
USB	Univerzální sériová sběrnice – Universal Serial Bus
XML	Extensible Markup Language
WSGI	Web Server Gateway Interface
Wi-Fi	Wireless Fidelity

Seznam příloh

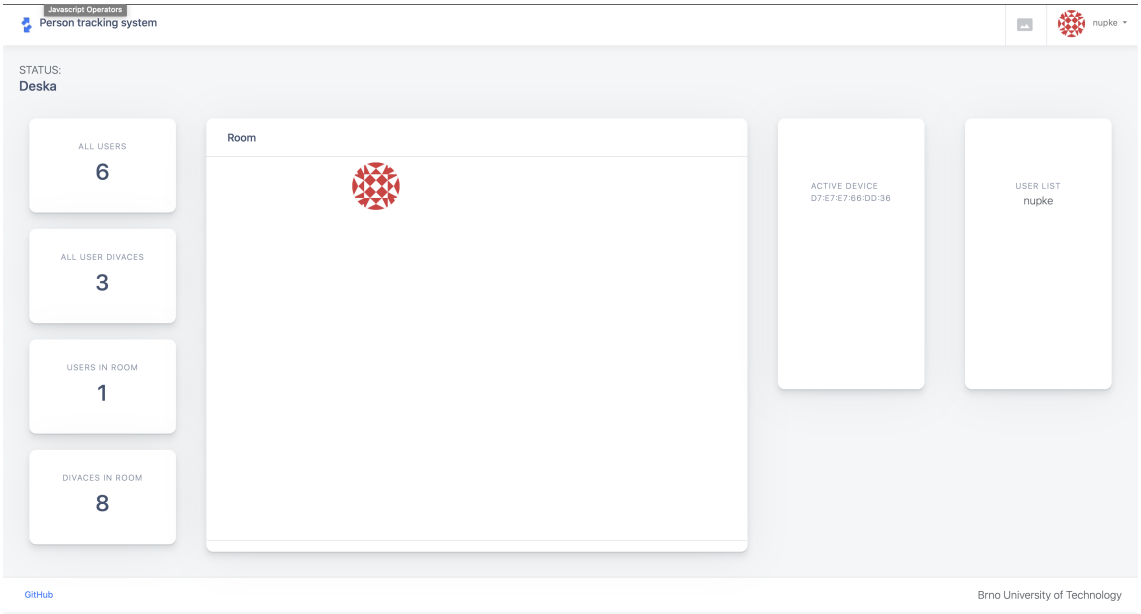
A Grafické prostředí aplikace



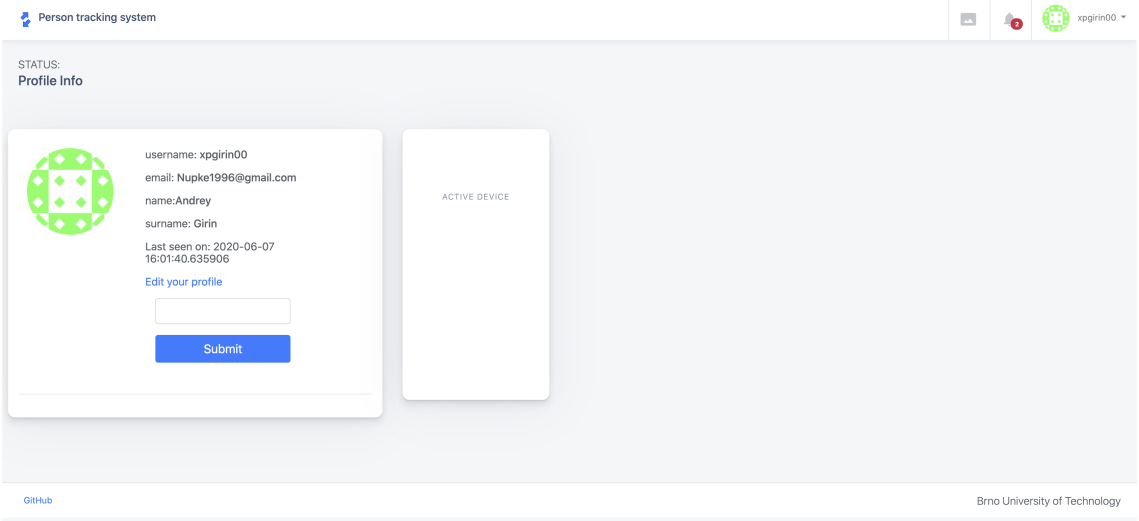
Obr. A.1: Registrační okno



Obr. A.2: přihlašovací okno



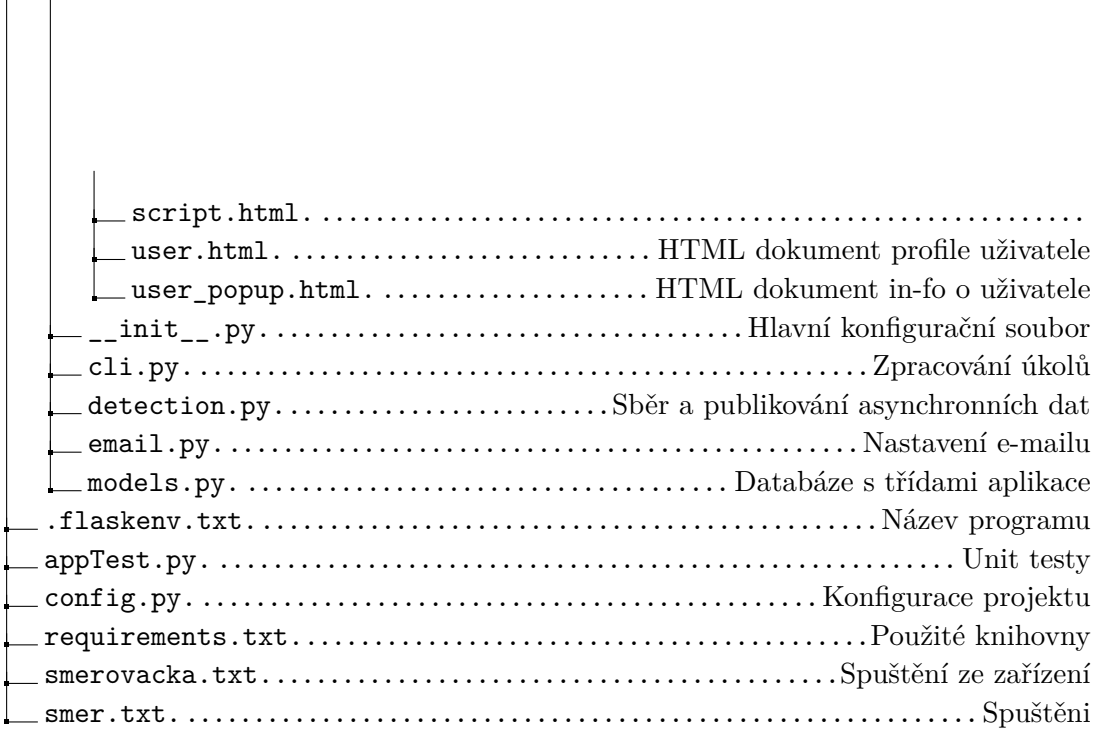
Obr. A.3: Hlavni stránka aplikace



Obr. A.4: Uživatelský profil

B Obsah přiloženého CD

- / ..... kořenový adresář přiloženého CD
- / ..... Bakalářská práce v v elektronické podobě
  - app .....
    - api .....
      - __init__.py ..... Blueprint api
      - auth.py ..... Api autentifikacija
      - errors.py ..... Zpracování chyb
      - tokens.py ..... Api generace tokenu
      - users.py ..... Api uživatelů
    - auth ..... Blueprint auth
      - __init__.py .....
      - email.py ..... Token obnovení hesla
      - forms.py ..... Formuláře k Registrace
      - routes.py ..... URL cesty autentifikace
    - errors c
      - __init__.py ..... Blueprint errors
      - handlers.py ..... Zpracování chyb
    - main ..... adresář hlavni stranky
      - __init__.py ..... Blueprint main
      - forms.py ..... Formuláře k vyplnění
      - routes.py ..... URL cesty aplikace
    - static ..... adresář k statickým souborům
      - css ..... adresář ze stylem
        - dashboards.css ..... CSS styly pro cely projekt
        - extras.1.1.0.min.css ..... CSS styly doplňková knihovna
      - images ..... adresář s obrázky
        - log.svg ..... logotyp aplikace
      - js ..... adresář s JavaScript souborem
        - Socket.js ..... JavaScript soubor s logikou aplikace
    - templates ..... adresář šablon aplikace
      - auth ..... adresář s přihlašovací stránkou
        - login.html ..... HTML dokument přihlašovací stránky
        - register.html ..... HTML dokument registrační stránky
        - reset_password.html ..... HTML dokument stránky obnovit heslo
        - reset_password_request.html .. HTML dokument stránky na dotaz obnovit heslo
      - email ..... adresář s emailovou zprávou
        - reset_password.html ..... HTML dokument pro zprávy na email
        - reset_password.txt ..... Textový dokument pro zprávy na email
      - 404.html ..... HTML dokument chyby 404
      - 505.html ..... HTML dokument chyby 505
      - base.html ..... HTML dokument pro základní internace
      - edit_profile.html ..... c nastaveni profilu
      - index.html ..... HTML dokument hlavní stránky


script.html
user.html
HTML dokument profile uživatele
user_popup.html
HTML dokument in-fo o uživatele
__init__.py
Hlavní konfigurační soubor
cli.py
Zpracování úkolů
detection.py
Sběr a publikování asynchronních dat
email.py
Nastavení e-mailu
models.py
Databáze s třídami aplikace
.flaskenv.txt
Název programu
appTest.py
Unit testy
config.py
Konfigurace projektu
requirements.txt
Použité knihovny
smerovacka.txt
Spuštění ze zařízení
smer.txt
Spuštěni